**SQL queries**

**What is the main difference between ‘BETWEEN’ and ‘IN’ condition operators?**

BETWEEN operator is used to display rows based on a range of values in a row whereas the IN condition operator is used to check for values contained in a specific set of values.

 Example of BETWEEN:

SELECT \* FROM Students where ROLL\_NO BETWEEN 10 AND 50;

Example of IN:

SELECT \* FROM students where ROLL\_NO IN (8,15,25);

**# Want to add Count of each color in separate column ## analytics function**

For example : Table : Bricks

|  |  |  |  |
| --- | --- | --- | --- |
| **Brick\_ID** | **Color** | **Shape** | **Weight** |
| 1 | Blue | Cube | 1 |
| 2 | Blue | Pyramid | 2 |
| 3 | Red | Cube | 1 |
| 4 | Red | Cube | 2 |
| 5 | Red | Pyramid | 3 |
| 6 | Green | Pyramid | 1 |

Select b.\*, (select \* from Bricks where color=b.color) Total\_bricks\_by\_color from Bricks b

Or

Select b.\*, count(\*) over(partition by color) Total\_bricks\_by\_color from Bricks b

**Output :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Brick\_ID** | **Color** | **Shape** | **Weight** | **Total\_bricks\_by\_color** |
| 1 | Blue | Cube | 1 | 2 |
| 2 | Blue | Pyramid | 2 | 2 |
| 3 | Red | Cube | 1 | 3 |
| 4 | Red | Cube | 2 | 3 |
| 5 | Red | Pyramid | 3 | 3 |
| 6 | Green | Pyramid | 1 | 1 |

**## In case you want sum of weight by Shape**

Select b.\*, sum(weight) over(partition by shape) Total\_weight\_by\_shape from Bricks b

**Another examples :**

Now we want to show all the employees’ salaries along with the highest salary by job title.

The query is very similar to the previous one. The only two changes are the aggregate function and the column in PARTITION BY.

|  |
| --- |
| SELECT first\_name,       last\_name,       job\_title,       department,       salary,       MAX(salary) OVER (PARTITION BY job\_title) AS max\_salary\_by\_job\_title  FROM employees; |

**3rd Highest salary:**

It is also important to note that the LIMIT clause takes in 2 components. The first component refers to the number of rows that we need to skip from the top. The second component refers to the number of rows that is to be displayed.

SELECT emp\_name AS Employee, salary AS Salary FROM employee ORDER BY salary DESC LIMIT 2,1;

SELECT \* FROM(

SELECT emp\_name, salary, RANK()

over(ORDER BY salary DESC) AS ranking FROM employee) AS k

WHERE ranking=3;

SELECT \* FROM(

SELECT emp\_name, salary, ROW\_NUMBER()

over(ORDER BY salary DESC) AS ranking FROM employee) AS k

WHERE ranking=3;

**To Create Temporary Table:**

CREATE TABLE #EmpDetails (id INT, name VARCHAR(25))

Given a table of candidates and their skills, you're tasked with finding the candidates best suited for an open Data Science job. You want to find candidates who are proficient in Python, Tableau, and PostgreSQL.

Write a query to list the candidates who possess all of the required skills for the job. Sort the output by candidate ID in ascending order.

**Answer :** SELECT candidate\_id FROM candidates where

skill in ('Python','Tableau','PostgreSQL')

GROUP BY candidate\_id having count(skill) > 2 ORDER BY candidate\_id;

Assume you're given the table on user viewership categorised by device type where the three types are laptop, tablet, and phone.

Write a query that calculates the total viewership for laptops and mobile devices where mobile is defined as the sum of tablet and phone viewership. Output the total viewership for laptops as laptop\_reviews and the total viewership for mobile devices as mobile\_views.

**Answer:** SELECT

(SELECT COUNT(\*) from viewership WHERE device\_type = 'laptop') AS laptop\_views,

(SELECT COUNT(\*) from viewership WHERE device\_type in ('phone','tablet')) as mobile\_views;

**# SQL query to get Duplicate records in table**

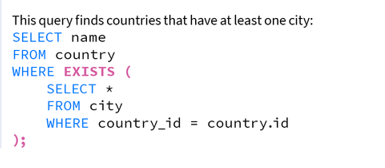
SELECT column1, column2, ..., COUNT(\*) as count

FROM table\_name

GROUP BY column1, column2, ...

HAVING COUNT(\*) > 1;

**# Correlated Query & self join**



**# Nested Queries ( Query inside query )**

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within clauses, most commonly in the WHERE clause

Select \* from ( select \* from table ) ~ select \* from table

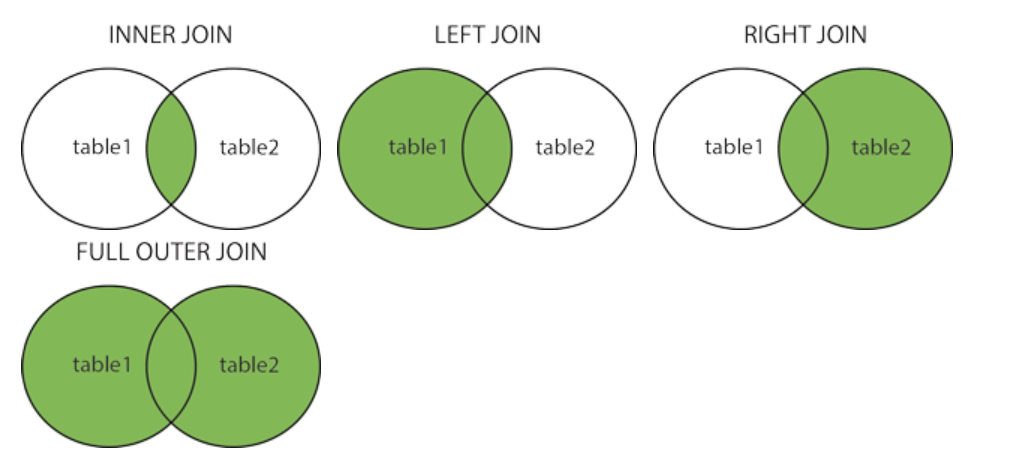
Select \* from emp where deptno = ( select deptno from dept where loc=’New York’) **## This will accept only one single return value. In order to accept more values use “IN” operator instead of “=”**

Select job, ename, (Select \* from dual) from from emp --> **This will return Output as Sub-query has returning only one value and If sub-query here is giving single value then we’re good to run else it’ll throw error “single-row subquery returns more”**

Select job, ename, (Select “Hello” from dual) from from emp --> **This will have additional column in Output as Hello**

## Different Types of SQL JOINs

* (INNER) JOIN: **Returns records that have matching values** in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



Select e.emp, e.job, e.sal from emp e, dept d

Where e.deptno=d.deptno

And d.loc=’dallas’ and e.job=’Manager’

Or

Select e.emp, e.job, e.sal from ( select \* from emp where job=’Manager’) e, (select \* from dept where loc=’dallas’) d Where e.deptno=d.deptno

**# Self Join**

Select e.ename as “Employee, m.ename as “manager”

From emp e, emp m where e.mgr=m.empno

Or

Select e.ename as “Employee, m.ename as “manager”

From emp e INNER Join emp m on e.mgr=m.empno

**# Natural Join**